Retrieval and Adaptation of Cases Using an Artificial Neural Network

Ricardo B. Sovat André C. P. L. F. de Carvalho

Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, SP, Brazil
sovat, andre@icmc.sc.usp.br

Abstract

One of the main issues in the research of Case-Based Reasoning systems is the retrieval and adaptation of cases. This paper proposes the use of Artificial Neural Networks for the retrieval and adaptation steps of a Case-Based Reasoning cycle. Its main goal is to handle cases in the domain of Artificial Neural Networks models design. Moreover, it addresses questions related to the attributes relevance and the coding of the inputs and outputs of an Artificial Neural Network.

Introduction

This paper proposes a mechanism, based on Artificial Neural Networks (ANNs) for the retrieval and adaptation of cases. This mechanism was necessary due to the nature of the cases treated by the Case-Based Reasoning (CBR) system designed by the authors. Each case represents an ANN designed by an expert to solve a particular problem. This work deals with the problem of selecting and adapting these cases. These cases can be used to support the design of ANN to deal with new problems.

The reason behind the use of CBR for the design of ANNs is the lack of a clear set of rules to select the free parameters of ANNs. In spite of the growing use of ANNs in the design of intelligent systems, there is a lack of a well established knowledge on how to obtain the better performance or the best fit between an specific model and a given task (Haykin 1994). The approaches used are basically empirical. A novice user usually has difficulties to design an efficient ANN to solve a given problem.

During the development of the case base, the authors observed that most of the cases presented a relatively small case section followed by a larger solution part. Besides, similar to the way a specialist works, the initial selection of a paradigm, even being important, was less decisive than the tuning of its several parameters. In this tuning process, however, the correct setting of initial values seems to have a great influence.

Inside the CBR approach (Kolodner 1993), this means that the adaptation step plays a major role in the process. A large amount of the time would be spent adjusting the parameters of the proposed final model. This kind of adjustment requires, usually, either the direct user's interference or the firing of a sequence of rules, inside an environment where the knowledge is not well structured. The adaptation of cases is one of the main open problems in the design of CBR systems. As a result, it was decided to test the performance of an ANN working at the same time as an indexing and an adaptation method.

This paper is organized as follows: the next section makes a brief reference to previous related works, as well as the placing of the proposed system inside suggested taxonomies. After this, another section will show how the case base was constructed. The features taken in account when creating the ANN model chosen for perform the indexing/adaptation task and its basic mechanism are presented in the last section.

Related Works

The use of ANNs inside a CBR cycle (Aamodt and Plaza1994) has been proposed by many research groups: (Shin and Park 2000), (Corchado and Lees 2000), (Malek 2000), (Fabiunke et al. 1997), (Domeshek 1993), (Petridis and Paraschidis 1993), (Sase et al. 1993), (Wendel 1993), (Becker and Jazayeri 1989) and (Thrift 1989). This can be seen not only as a research issue, but also as a natural tendency in CBR methodology: the integration of CBR with other intelligent techniques.

According to these previous works, the integration can be achieved either through the division of tasks between the case-based reasoning system and the neural network or by the design of an intelligent architecture combining neural networks and case-based reasoning features. For the first approach there are, according to Reategui and Campbell (Reategui and Campbell 1994), four alternatives which generalises the different paradigms used:

- **Central Control**, where the case-based reasoning and the neural network are controlled by a central device;
- **Distributed Control**, where the control is divided between the two techniques;
- **Neural network dominant**, where the control is biased towards the neural network.
- **Case-based reasoning dominant**, where the control is biased towards the case-based reasoning component;

Copyright © 2001 Ricardo Sovat, André Carvalho

The work described in this paper belongs to the last of these alternatives.

Also, according to Hilario (Hilario 1997), the proposed architecture is defined as a hybrid approach (combine neural networks with symbolic models), sub-classified as functional (incorporate complete symbolic and connectionist components) with a coprocessing integration mode (the two components are equal partners in problem solving process, each interacting directly with the environment and receiving information one from the other).

The Case Base

In order to capture the main features taken into consideration by a specialist or an expert user during the selection and configuration of an ANN model, three kinds of information were selected:

- task type;
- data profile;
- environment available.

The information about the task involve essentially the kind of application:

- classification (including clustering);
- regression (forecasting or function adjusting)
- optimization and the data format (numerical, symbolic or both).

According to the task, specific information can be required, e.g., the number of classes, the number of sample features (sample attributes) or the look ahead distance. This request can be introduced using preprocessing rules or adjusting the weights of the respective descriptive attributes. For an initial experiment, only classification tasks are considered.

The data profile is obtained from a set of statistical measures taken from the correspondent dataset. The initial measures set is influenced by the fact that the first task type to be examined is classification, but future expansions are being studied. It comprehends the total amount of samples, the ratio between the standard deviation inside each class and the standard deviation of the whole dataset, the coefficient of excess (kurtosis) and the coefficient of skewness of the dataset.

These measures are calculated as an average of all the continuous attributes of the dataset to be classified. At this point, two different ideas are being considered : to expand the coefficients to discrete variables, using an adequate formula, or to keep the discrete values out of the average calculation, introducing an extra attribute expressing the total amount of discrete variables.

The statistical measures aim to provide an indication of how easy or difficult will be the ANN training. There is a simple measure (amount of samples) that is fundamental, once it denotes the basic power of the training phase. The other three measures try to express the general shape of the data distribution. For each class, the farther the standard deviation ratio gets from 1.0, the higher the chance that this class characterization will be different of the whole dataset behavior, suggesting it will be more or less easily learned. The last two coefficients try to give an idea of the proximity of the data to the normal distribution.

The coefficient of excess shows the distribution flatness, indicating the presence of relevant data even far from the mean. The skewness coefficient reveals asymmetry between the classes distribution, with a class being more present than others theoretically making the training harder.

Other measures could also be added, including values of correlation between the attributes or features commonly used by theory of information analysis, as the data entropy. However, the idea is to keep the measures set minimal, in order to allow hints of the dataset shape without having to perform expensive calculation, sometimes not available in the most common statistical packages.

Not only the network parameters will be influenced by these selected measures, but also the choice of the paradigm. Data presenting a low level of error rate, or a high degree of separation between classes, will be satisfied by simpler models.

Originally, environment data aim to provide information about possible restrictions or lack of computing resources that can be faced by the user. It includes, basically, the computer used, the time available for the training of the ANN and the user’s experience. Its main influence takes place in the training error tolerance required. With the continuous increase in the computational power of personal computers and since the final system seems to be more aimed to novice users, these fields probably are going to always receive very similar values, thus decreasing its discrimination power and tending to not be taken in account during the indexing phase.

Since the goal of the work is to test the performance of an indexing mechanism, the case base is being built as a flat memory. This will simplify its construction, consequently turning the search for the best case more difficult. The attributes describing the desired task will be used as indexes, if necessary, to speed up the retrieve step.

In this context, two steps are planned for the adaptation phase. The first step will be accomplished by the ANN. It includes any kind of preprocessing needed to conform the user case in order to ease the matching with the base cases. It also includes initial adaptation operations that complete the case retrieved by adding any attribute needed which is not in the base case.

Due to the variety of possible solutions, it is common to have a good solution that does not have all the parameters needed to configure the final model. Many parameters are unknown or not informed by the original designer. Even being a good solution, for this case, some attributes were considered irrelevant. If those attributes were left for the user to choose, the design process would became more difficult. The ANN, after performing the selection of the

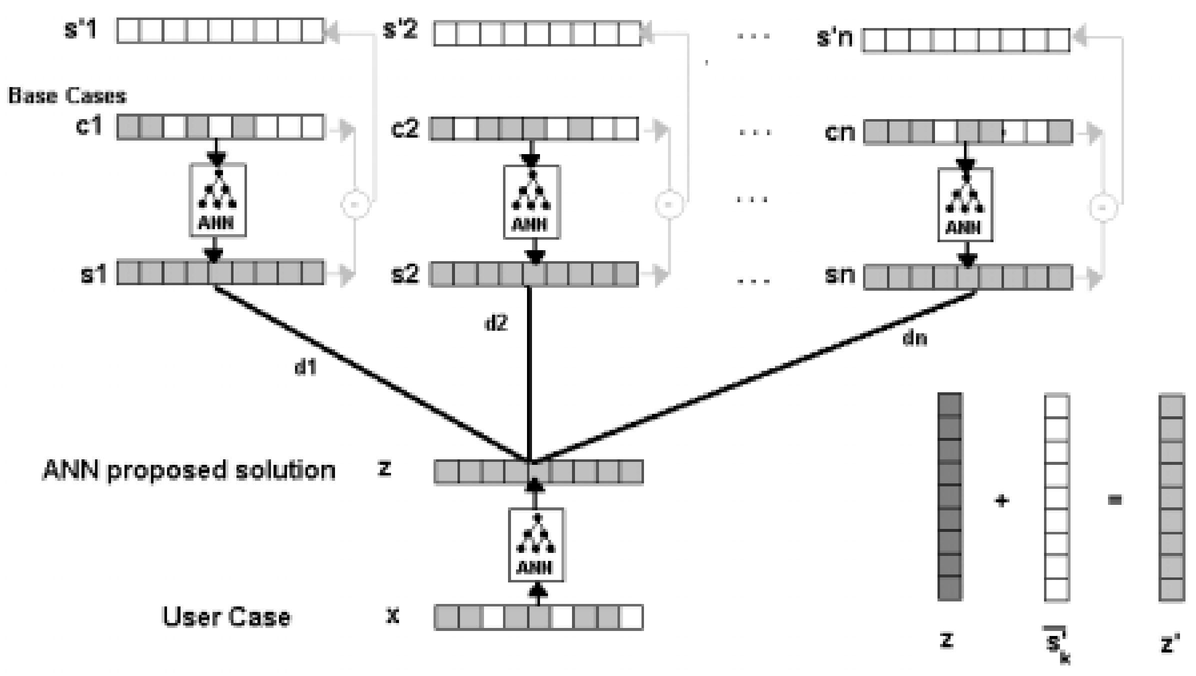

Figure 1 - Associative Memory Overall Functioning

best case, executes an initial parameters completion, somehow introducing an implicit model, without having to make it explicit by using rules.

The final adaptation, however, deals with the transformation part of the process and is accomplished by rules. This kind of adaptation does not add (or remove) any attribute, but it changes the values in the current attributes. These changes are related mainly to the network topology. This kind of parameter differs from the rest, once they can be determined reasonably well, using algebraic formulae. The CBR shell employed to test the base is able to represent both kinds of adaptation. It builds the case base from a script where all the elements previously mentioned can be specified. Beside them, some generalizations will also be included. When matching the continuous values, as in the statistical measures, values ranges will be defined, in order to ease the matching process.

Using the ANN to Retrieve a Case

The basic idea in introducing an ANN into the retrieving step is to implement a pattern recognition method able to correctly identify similarities between two attribute vectors. In order to perform an initial adaptation, this method needs to complete the pattern recognized, so that a full vector can be obtained. The solution proposed to this problem is to use an associative memory. This model will be trained to associate each case in the case base to a vector of constant length. Initially, the ANN employed will be a MLP network trained with the backpropagation algorithm (Haykin 1994), to start with a well known implementation. However, other associative models are intended to be tested, like an ART2 network (Carpenter and Grossberg 1987).

The ANN input layer will have as many units as necessary to code the attributes of the stored cases. In order to represent the cases inside the CBR shell, those attributes can be expressed in one of five types :

- numerical (real or integer);
- enumerated (discrete);
- lists;
- strings;
- booleans.

Numerical and boolean attributes use one unit each to be represented, directly in a numerical format. Enumerated attributes have to be first encoded by an algorithm that takes in account the existence or not of internal similarities, as it happens when there is an order inside the enumeration, for example. This information is included in the base, next to the enumerated attribute definition. Usually, enumerated attributes need as many units as the cardinality of the enumeration.

Lists attributes can contain any of the other types, and they will be handled in the same way as if they are not list members. In this first version, string attributes are not being used as inputs, just as auxiliary information.

After the conversion and normalization of the attributes, they are presented to the ANN's input layer. The ANN output layer has a fixed number of units divided in two areas. The first area is composed of eight units and always represents the same solution attribute group: the ANN model (MLP, RBF, SOM), the sizes of the test and validation subsets, the number of epochs between validations, the activation function, the initialization function, the updating function and the learning rate (learning height for SOM's).

Next to this area, there is a five units group that holds values that can vary in meaning. It is important to notice that, for the associative memory training, these meanings make no difference, because there is no meaning context to the ANN. After the association, rules are used to transform the numbers in the respective parameters, according to the chosen model. Thus, for example, the first unit of this group can indicate parameters like the momentum rate in a MLP model or the learning radius in a SOM model.

The hidden layer is being kept with a size (number of units) between the number of attributes of the stored cases and the quantity of output units. This size itself will be determined empirically or obtained after a pruning method.

In spite of the advantages expected, employing an ANN as the indexing method presents two new problems. The first one is the need of periodically retraining the ANN. This happens because the network learns the initial case base, presented when it was first trained. As the system begins to learn new cases, this mechanism tends to became inaccurate. The second problem comes from the inability of a simple ANN model in ranking solutions. The ANN can choose a case from the base, but it can not provide a second or third best suggestions.

In order to find a solution to both problems, the authors adopted a model of associative memory inspired by Tetko in (Tetko 2001). In this model, two methods work in parallel: a set of ANNs (in this case just one ANN executed many times) and a nearest neighbor (k-NN) algorithm.

The k-NN result is used for two goals : to adjust the ANN's output and to provide a optional ranking, in the case of the first selected case not be approved by the user.

The associative memory overall functioning is illustrated in Figure 1.

Following the suggestions proposed by the associative memory used, the ANN output vector **z** obtained from the user case **x** may not be the final value, requiring adjustments. Thus, a group of **n** cases (selected initially using the base index definition, through the k-NN algorithm) is also inputted to the ANN, generating a set of vectors **s**. Since these cases were used to train the ANN, the correspondent output vectors should be equal to the solutions stored in the case base. This would always happen if the ANN was updated and able to work without introducing errors. When this does not occur, any discrepancies are stored in a set of vectors **s'**.

In order to obtain these vectors, continuous variables simply have their differences calculated. Discrete attributes are submitted to a brief adjust to take into account the internal proximity.

At this point, the k-NN algorithm is able to obtain its ranking of the cases, by calculating the distance **d** between each vector **s** and the vector **z**.

However, in order to achieve the best adjustment to the initial proposed solution, the final ANN proposed solution will be the vector **z'**. This vector is the sum of the vector **z** with a vector $\mathbf{s'}_{\mathbf{k}}$, which contains the arithmetic average of the **k** nearest vectors **s** belonging to the k-NN ranking..

The correction provided by the k-NN algorithm creates a second level of knowledge, perhaps better described as a local knowledge, complementing the global knowledge stored in the ANN. The ANN can make a first selection. This selection will be as good as the proximity of the current case base to the original one used for the training of the ANN. This method compensates the problem of the ANN outdating.

As the case base grows and the ANN is kept without training, the k-NN correction becomes larger. A value can be established for this correction so that it forces a new training of the ANN, but it does not have to be executed so often. Even when the ANN is recently trained, the k-NN adjustment is expected to be more influent as the case base gets larger, due to the consequent higher complexity of the solution space and the possible decrease in the ANN's generalization power.

Besides, the case ranking constructed by the k-NN algorithm can be used to reselect cases if necessary. Each case in the ranking was presented to the ANN input layer, (furnishing an output vector **s**) and can continue the CBR cycle. As an initial approach, the k-NN algorithm implemented in the shell uses Euclidean distances as the similarity measure.

To properly process an input, the ANN input layer representation has also to handle the problem of missing values in the case base. These missing values come from attributes whose values were not defined by the user.

The encoding algorithm handles these values in two different ways: 'unknown' and 'undefined' attributes. Attributes not present in the case or assigned as 'not applicable" by the user are treated equally: they are regarded as not influencing the selection and considered undefined. Attributes marked as 'unknown' are managed differently. They are considered as existent and influent.

The 'unknown' values are substituted by the arithmetic average (or mode, for discrete fields) of all values of that attribute appearing in the case base, both in the k-NN algorithm and the ANN input layer.

The 'undefined' (or 'not applicable') attributes cause the k-NN algorithm to make their distances equal to zero. However, the ANN cannot do the same. Such inputs have to receive another neutral value, once zero is significant to the algorithm. This value is the arithmetic average (or the mode) obtained from the base attribute values, but only considering those appearing in a case together with values equal to the actually provided by the user. Occurrences of an attribute not being observed at the same time of those appearing in the user case are not taken into account.

Conclusion

At this moment, this work is in the beginning of the case base construction, what does not allow a representative benchmark to determine its performance.

However, the development environment used allows to perform tests with the retrieval and the adaptation phases of the CBR cycle, both using or not the proposed ANN associative memory. If the ANN is disabled, it is possible to work only with rules and the k-NN algorithm. This will be used to compare both approaches.

In order to measure the quality of the solution provided by the system as a whole, the ANN models proposed by the CBR methodology will be compared with those constructed by human experts.

It is important to stress that the adoption of an ANN inside the CBR cycle aims mainly the achievement of a better implementation of the adaptation step. Nevertheless, there is also a great interest in the study the possible alternatives for the hybridization among the connectionist approaches and the traditional nearest neighbor methods employed during the indexing and retrieving.

As future works, beside the immediate increase of the case base and the performance tests, the authors plan the experimentation with other types of associative memory, and the selection of alternative k-NN distance measures. On the final system itself, it is intended to diversify the case base, with the inclusion of cases related to other task types, the expansion of the ANN models covered and the incorporation of new case and solution attributes.

Acknowledgments

The authors would like to thank the support received from FAPESP, the São Paulo State research agency

References

Aamodt, A. and and Plaza, E. 1994. Case-based reasoning: foundational issues, methodological variations, and system approaches. *Artificial Intelligence Communications*, 7, 39-59, March.

Becker, L. Jazayeri, K. 1989. A Connectionist Approach to Case-Based Reasoning. *DARPA Workshop on Case-Based Reasoning*, Hammond, K. J. Pensacola Beach, USA, 213-217. Morgan Kaufmann, May-June.

Carpenter, G. and Grossberg, S. 1987. "ART 2: self-organization of stable category recognition codes for analog input patterns". *Applied Optics*, 26, 4919—4930, 1987.

Corchado, J. M., and Lees, B., 2000 Adaptation of Cases for Case Based Forecasting with Neural Network Support. In Soft Computing in Case Based Reasoning, chapter 13. Pal, S. K., Dillon, T. S., Yeung, D. S., eds. London, England : Springer-Verlag.

Domeshek, E. 1993. A case study of case indexing: designing index feature sets to suit task demands and support parallelism. *Advances in Connectionist and Neural Computation Theory*, 2*: Analogical Connections*, Bamden, J. and Holyoak, K.. Norwood, USA. Ablex.

Fabiunke, M. Kock, G. Milaré, C. and Carvalho, A. 1997. A Hybrid System Integrating Neural Network and Case-Based Reasoning Features. *Technical Report*, ISSN-0103-2569, Institute of Mathematical Sciences and Computing, University of São Paulo. São Carlos, Brazil.

Haykin, S. 1994. Neural Networks. A Comprehensive Foundation. Macmillan College Publishing Company. Hamilton, Canada.

Hilario, M., 1997 *An Overview of Strategies for Neurosymbolic Integration*. In Connectionist Symbolic Integration, chapter 1. Hillsdale, NJ : Lawrence Erlbaum,.

Kolodner, J. 1993. *Case-based reasoning*. Morgan Kaufmann.

Malek, M., 2000 *Hybrid Approaches for Integrating Neural Networks and Case Based Reasoning: From Loosely Coupled to Tightly Coupled Models*. In Soft Computing in Case Based Reasoning, chapter 4. Pal, S. K., Dillon, T. S., Yeung, D. S., eds. London, England : Springer-Verlag.

Petridis, V. and Paraschidis, K. 1993. Structural adaptation based on a simple learning algorithm. *International Joint Conference on Neural Networks*, 1, 621-623, Baltimore, USA.

Reategui, E. and Campbell, J. 1994. A classification system for credit card transactions. *European Workshop on Case-Based Reasoning*, Keane, M., Haton, J. and Manago, M. 167-174, Chantilly, France.

Sase, M., Matsui, K. and Kosugi, Y. 1993. Inter-generational architecture adaptation of neural networks. *International Joint Conference on Neural Networks*, 3, 2941-2944, Baltimore, USA.

Shin, C-K, and Park, S. C., 2000 Towards Integration of Memory Based Learning and Neural Networks. In Soft Computing in Case Based Reasoning, chapter 5. Pal, S. K., Dillon, T. S., Yeung, D. S., eds. London, England : Springer-Verlag.

Tetko, I. V., 2001 *Associative Neural Network*, CogPrints archive code: cog0000144, http://cogprints.soton.ac.uk.

Thrift, P. 1989. A neural network model for case-based reasoning. *Workshop on Case-Based Reasoning*, Hammond, K. Pensacola Beach, USA, 334-337, Morgan Kaufmann, May-June.

Wendel, O. 1993. Case based reasoning in a simulation environment for biological neural networks. *First European Workshop on Case-Based Reasoning*. Wess, S., Althoff, K. and Richter M., University of Kaiserslauten, Kaiserslauten, Germany, 1, 1-5.